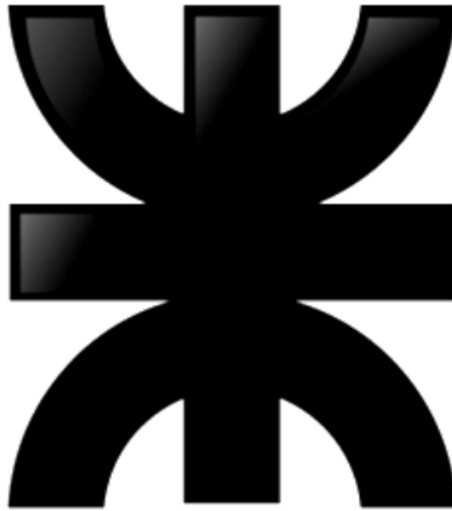


UNIVERSIDAD TECNOLÓGICA NACIONAL

FACULTAD REGIONAL RESISTENCIA



SISTEMAS DE GESTIÓN DE BASES DE DATOS

Trabajo Práctico N°5: Datos semiestructurados

Equipo de Cátedra:

- Profesor: I.S.I. Andrés Pablo Fantín
- J.T.P.: I.S.I. Juan Carlos Fernández
- Auxiliar Adscripta: Luciana Campestrini

Alumnos:

- Cristaldo, Cristian
- Maurel Garcete, Philippe
- Piragine, Tomás
- Fernandez Bruno Ulises

1) Escribir un documento **XML** para registrar préstamos de una biblioteca.

a. En el documento se indicarán:

- i. El nombre y apellidos del bibliotecario responsable del préstamo.
- ii. Fecha del préstamo y de devolución.
- iii. Datos del lector (id, nombre, apellidos, teléfono y dirección).
- iv. La dirección se dividirá en tipo de calle (que puede ser calle o avenida), nombre calle, número, piso y letra, código postal, localidad y provincia
- v. Un máximo de tres ejemplares en préstamo. Para cada uno de ellos: el número de registro, título, autor(es)
- vi. El préstamo tendrá un atributo numérico que servirá como identificador

```
<?xml version="1.0" encoding="UTF-8"?>
<prestamo id="12345">
  <bibliotecario>
    <nombres>Laura</nombres>
    <apellidos>Gómez Fernández</apellidos>
  </bibliotecario>
  <fechas>
    <fechaPrestamo>2024-10-21</fechaPrestamo>
    <fechaDevolucion>2024-11-21</fechaDevolucion>
  </fechas>
  <lector>
    <id>001234</id>
    <nombre>Carlos</nombre>
    <apellidos>Martínez Pérez</apellidos>
    <telefono>123-456-789</telefono>
    <direccion>
      <tipoCalle>Calle</tipoCalle>
      <nombreCalle>Belgrano</nombreCalle>
      <numero>25</numero>
      <piso>3</piso>
      <letra>B</letra>
      <codigoPostal>28013</codigoPostal>
      <localidad>Sáenz Peña</localidad>
      <provincia>Chaco</provincia>
    </direccion>
  </lector>
  <ejemplares>
    <ejemplar>
      <numRegistro>1001</numRegistro>
      <titulo>Patrones de Diseño</titulo>
      <autores>
        <autor>Niklaus Wirth</autor>
      </autores>
    </ejemplar>
    <ejemplar>
      <numRegistro>1002</numRegistro>
      <titulo>Cálculo Diferencial e Integral</titulo>
      <autores>
        <autor>Martin Fowler</autor>
      </autores>
    </ejemplar>
  </ejemplares>
</prestamo>
```

```
<numRegistro>1003</numRegistro>
<titulo>Guía de Usuario UML</titulo>
<autores>
  <autor>Carlos Pineda</autor>
</autores>
</ejemplar>
</ejemplares>
</prestamo>
```

b. Verificar que este bien formado.

El documento **XML** está bien formado porque cumple las siguientes reglas:

1. Los elementos deben seguir una estructura de árbol (estrictamente jerárquica). Es decir un solo nodo raíz.
2. Los elementos deben estar correctamente anidados.
3. Los elementos no se pueden superponer entre ellos.
4. Los documentos deben tener un nodo raíz.
5. Todas las etiquetas deben estar debidamente cerradas.
6. Las etiquetas vacías (etiquetas sin contenido) deben tener una sintaxis especial.
7. Un nombre de elemento, atributo, entidad, etc., comienza por una letra, y continúa con letras, dígitos, guiones, rayas, punto, dos puntos.
8. No pueden utilizarse las palabras XML, xml, Xml, etc., como caracteres iniciales del nombre de un atributo, entidad, etc.
9. XML es sensitivo a mayúsculas y minúsculas (no es lo mismo <Autor> que <autor>)
10. El uso de espacios en blanco, y los saltos de línea, funcionan al igual que en HTML (sólo se toma en cuenta cuando aparece en el valor de un atributo, o cuando se indica su grado de significado).

liquid-technologies.com/online-xml-validator

Free Online XML Validator (Well formed)

Validates that an XML document is well formed, if you have a schema use the appropriate validator instead (XSD, RelaxNG or Schematron).

liquid Complete XML Toolkit

Access the online tools directly from your desktop.
Download Free Liquid Studio Community Edition Now!

XML data to validate

```
39      <autor>Martin Fowler</autor>
40    </autores>
41  </ejemplar>
42  <ejemplar>
43    <numRegistro>1003</numRegistro>
44    <titulo>Guía de Usuario UML</titulo>
45    <autores>
46      <autor>Carlos Pineda</autor>
47    </autores>
48  </ejemplar>
49  </ejemplares>
50 </prestamo>
51
```

Validate

Document Valid

2) De acuerdo al siguiente documento **XML**, realizar lo siguiente:

Documento XML de catalogo de películas:

<?xml version="1.0"?>

<CatalogoPelículas>

<Película>

<Titulo>The Matrix</Titulo>

<Duracion>136</Duracion>

<Genero>Sci-Fi and Fantasy</Genero>

<Actores>

<Actor>Keanu Reeves</Actor>

<Actor>Laurence Fishburne</Actor>

<Actor>Carrie Ann Moss</Actor>

</Actores>

<Fecha>1999</Fecha>

<Director>Wachowski Brothers</Director>

<Formato>DVD</Formato>

</Película>

<Película>

<Titulo>Titanic</Titulo>

<Duracion>194</Duracion>

<Genero>Drama</Genero>

<Actores>

<Actor>Leonardo DiCaprio</Actor>

<Actor>Kate Winslet</Actor>

</Actores>

<Fecha>1999</Fecha>

<Director>James Cameron</Director>

<Formato>DVD</Formato>

</Película>

<Película>

<Titulo>The Sixth Sense</Titulo>

<Duracion>106</Duracion>

<Genero>Thriller</Genero>

<Actores>

<Actor>Bruce Willis</Actor>

<Actor>Haley Joel Osment</Actor>

</Actores>

<Fecha>1999</Fecha>

<Director>M. Night Shyamalan</Director>

<Formato>VHS</Formato>

</Película>

</CatalogoPelículas>

a) Verificar que sea un documento **bien formado**

1. Los elementos deben seguir una estructura de árbol con un solo nodo raíz.
 - Se verifica ya que el único nodo raíz es **<CatalogoPeliculas>...</CatalogoPeliculas>**.
2. Los elementos deben estar correctamente anidados.
 - Se verifica ya que todos los elementos cumplen con dicha condición.
3. Los elementos no se pueden superponer entre ellos.
 - Efectivamente los elementos no se superponen entre ellos, respetando su orden.
4. Todas las etiquetas deben estar cerradas.
 - Podemos observar que todas las etiquetas se encuentran debidamente cerradas.
5. Un nombre de elemento, atributo, entidad, etc., comienza por una letra, y continúa con letras, dígitos, guiones, rayas, punto, dos puntos.
 - Todos los nombres de elementos y atributos comienzan con una letra.
6. No pueden utilizarse las palabras XML, xml, Xml, etc., como caracteres iniciales del nombre de un atributo, entidad, etc.
 - Se verifica que ningún nombre de atributo, entidad o elemento comienza con las palabras "XML", "xml", "Xml", etc.
7. XML es sensitivo a mayúsculas y minúsculas.
 - Se verifica que para cada elemento, en su marca de inicio y de fin sean case sensitive.

b) Crear el documento **DTD** respectivo

```
<!DOCTYPE CatalogoPelicula SYSTEM "CatalogoPeliculas.dtd">
<!ELEMENT CatalogoPelicula (Pelicula+)>
<!ELEMENT Pelicula (Titulo, Duracion, Genero, Actores+, Fecha, Director, Formato)>
<!ELEMENT Titulo (#PCDATA)>
<!ELEMENT Duracion (#PCDATA)>
<!ELEMENT Genero (#PCDATA)>
<!ELEMENT Actores (Actor)>
<!ELEMENT Fecha (#PCDATA)>
<!ELEMENT Director (#PCDATA)>
<!ELEMENT Formato (#PCDATA)>
<!ELEMENT Actor (#PCDATA)>
```

c) Validar **XML** con el **DTD** creado.

```
<?xml version="1.0" encoding="UTF-8"?>
<CatalogoPelículas>
  <Película>
    <Titulo>The Fast and the Furious</Titulo>
    <Duracion>107</Duracion>
    <Genero>Accion aventura y suspenso</Genero>
    <Actores>
      <Actor>Vin Diesel</Actor>
      <Actor>Paul Walker</Actor>
      <Actor>Michelle Rodríguez</Actor>
    </Actores>
    <Fecha>2001</Fecha>
    <Director>Rob Cohen</Director>
    <Formato>DVD</Formato>
  </Película>
</CatalogoPelículas>
```

3) Escribir un documento **XML** que pueda ser **validado** con el siguiente **DTD**:

```
<!ELEMENT Libro (Titulo, Contenido, Copyright)>
<!ELEMENT Titulo (#PCDATA)>
<!ELEMENT Contenido ((Capitulo+, Separacion?)+)>
<!ELEMENT Capitulo (Tema, Seccion+)>
<!ATTLIST Capitulo materia (XML|Java) "Java">
<!ELEMENT Tema (#PCDATA)>
<!ELEMENT Seccion (#PCDATA)>
<!ATTLIST Seccion apartados CDATA #REQUIRED dificil (si|no) "no">
<!ELEMENT Separacion EMPTY>
<!ELEMENT Copyright (#PCDATA)>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Libro>
  <Titulo>Libro de Java</Titulo>
  <Contenido>
    <Capitulo materia="Java">
      <Tema>Introducción</Tema>
      <Seccion apartados="1" dificil="no">Primera sección</Seccion>
      <Seccion apartados="2">Segunda sección</Seccion>
    </Capitulo>
    <Separacion/>
    <Capitulo materia="XML">
      <Tema>XML y su uso</Tema>
      <Seccion apartados="1" dificil="si">Introducción al XML</Seccion>
      <Seccion apartados="3">Validación de documentos</Seccion>
    </Capitulo>
  </Contenido>
  <Copyright>2023</Copyright>
</Libro>
```

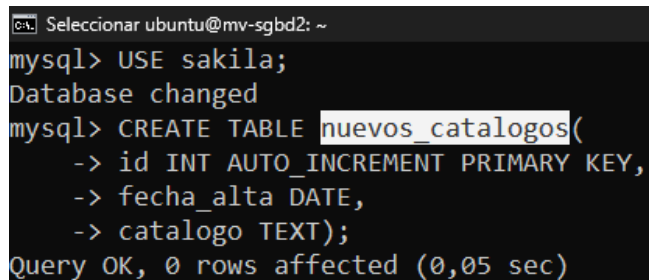
4) En la tienda de alquiler de películas se empezaron a recibir catálogos de manera más dinámica y de distintos proveedores, todos como documentos **XML**, lo que implica realizar algunas modificaciones a la base de datos. En la base de datos “**sakila**” realizar las siguientes acciones:

a. Crear una nueva **tabla** con el siguiente esquema:

- **nuevos_catalogos**(id: entero, fecha_alta: fecha, catalogo: texto)

```
USE sakila;
```

```
CREATE TABLE nuevos_catalogos(  
id INT AUTO_INCREMENT PRIMARY KEY,  
fecha_alta DATE,  
catalogo TEXT);
```



```
mysql> USE sakila;  
Database changed  
mysql> CREATE TABLE nuevos_catalogos(  
-> id INT AUTO_INCREMENT PRIMARY KEY,  
-> fecha_alta DATE,  
-> catalogo TEXT);  
Query OK, 0 rows affected (0,05 sec)
```


b. Agregar un registro a la tabla creada de manera de dar de alta el siguiente catálogo:

<CatalogoPelículas>

<Película>

<Titulo>The Matrix</Titulo>

<Duracion>136</Duracion>

<Genero>Sci-Fi and Fantasy</Genero>

<Actores>

<Actor>Keanu Reeves</Actor>

<Actor>Laurence Fishburne</Actor>

<Actor>Carrie Ann Moss</Actor>

</Actores>

<Fecha>1999</Fecha>

<Director>Wachowski Brothers</Director>

<Formato>DVD</Formato>

</Película>

<Película>

<Titulo>Titanic</Titulo>

<Duracion>194</Duracion>

<Genero>Drama</Genero>

<Actores>

<Actor>Leonardo DiCaprio</Actor>

<Actor>Kate Winslet</Actor>

</Actores>

<Fecha>1999</Fecha>

<Director>James Cameron</Director>

<Formato>DVD</Formato>

</Película>

<Película>

<Titulo>The Sixth Sense</Titulo>

<Duracion>106</Duracion>

<Genero>Thriller</Genero>

<Actores>

<Actor>Bruce Willis</Actor>

<Actor>Haley Joel Osment</Actor>

</Actores>

<Fecha>1999</Fecha>

<Director>M. Night Shyamalan</Director>

<Formato>VHS</Formato>

</Película>

</CatalogoPelículas>

```

SET @catalogo = '
<CatalogoPelículas>
  <Película>
    <Titulo>The Matrix</Titulo>
    <Duracion>136</Duracion>
    <Genero>Sci-Fi and Fantasy</Genero>
    <Actores>
      <Actor>Keanu Reeves</Actor>
      <Actor>Laurence Fishburne</Actor>
      <Actor>Carrie Ann Moss</Actor>
    </Actores>
    <Fecha>1999</Fecha>
    <Director>Wachowski Brothers</Director>
    <Formato>DVD</Formato>
  </Película>
  <Película>
    <Titulo>Titanic</Titulo>
    <Duracion>194</Duracion>
    <Genero>Drama</Genero>
    <Actores>
      <Actor>Leonardo DiCaprio</Actor>
      <Actor>Kate Winslet</Actor>
    </Actores>
    <Fecha>1999</Fecha>
    <Director>James Cameron</Director>
    <Formato>DVD</Formato>
  </Película>
  <Película>
    <Titulo>The Sixth Sense</Titulo>
    <Duracion>106</Duracion>
    <Genero>Thriller</Genero>
    <Actores>
      <Actor>Bruce Willis</Actor>
      <Actor>Haley Joel Osment</Actor>
    </Actores>
    <Fecha>1999</Fecha>
    <Director>M. Night Shyamalan</Director>
    <Formato>VHS</Formato>
  </Película>
</CatalogoPelículas>';

```

```

INSERT INTO nuevos_catalogos(fecha_alta, catalogo) VALUES (CURDATE(),
@catalogo);

```

```

ubuntu@mv-sgbd2: ~
mysql> SET @catalogo = '
'> <CatalogoPelículas>
'>   <Película>
'>     <Titulo>The Matrix</Titulo>
'>     <Duracion>136</Duracion>
'>     <Genero>Sci-Fi and Fantasy</Genero>
'>     <Actores>
'>       <Actor>Keanu Reeves</Actor>
'>       <Actor>Laurence Fishburne</Actor>
'>       <Actor>Carrie Ann Moss</Actor>
'>     </Actores>
'>     <Fecha>1999</Fecha>
'>     <Director>Wachowski Brothers</Director>
'>     <Formato>DVD</Formato>
'>   </Película>
'>   <Película>
'>     <Titulo>Titanic</Titulo>
'>     <Duracion>194</Duracion>
'>     <Genero>Drama</Genero>
'>     <Actores>
'>       <Actor>Leonardo DiCaprio</Actor>
'>       <Actor>Kate Winslet</Actor>
'>     </Actores>
'>     <Fecha>1999</Fecha>
'>     <Director>James Cameron</Director>
'>     <Formato>DVD</Formato>
'>   </Película>
'>   <Película>
'>     <Titulo>The Sixth Sense</Titulo>
'>     <Duracion>106</Duracion>
'>     <Genero>Thriller</Genero>
'>     <Actores>
'>       <Actor>Bruce Willis</Actor>
mysql>       <Actor>Haley Joel Osment</Actor>
mysql>     </Actores>
mysql>     <Fecha>1999</Fecha>
mysql>     <Director>M. Night Shyamalan</Director>
mysql>     <Formato>VHS</Formato>
mysql>   </Película>
mysql> </CatalogoPelículas>';
mysql> K, 0 rows affected (0,00 sec)

```

```

ubuntu@mv-sgbd2: ~
mysql> INSERT INTO nuevos_catalogos(fecha_alta, catalogo) VALUES (CURDATE(), @catalogo);
Query OK, 1 row affected (0,01 sec)

```

c. Utilizar las funciones **XML** disponibles en **MySQL** para:

i. **Obtener fecha de alta y actores** de la película **Titanic**.

El doble slash **//** se utiliza para buscar a través de la estructura **XML** los elementos que coincidan, sin importar su posición, lo que resulta más flexible.

```
SELECT
fecha_alta,
ExtractValue(catalogo, '//Pelicula[Titulo="Titanic"]/Actores/Actor/text()') AS actores
FROM nuevos_catalogos;
```

```
ubuntu@mv-sgbd2: ~
mysql> SELECT
-> fecha_alta,
-> ExtractValue(catalogo, '//Pelicula[Titulo="Titanic"]/Actores/Actor/text()') AS actores
-> FROM nuevos_catalogos;
+-----+-----+
| fecha_alta | actores |
+-----+-----+
| 2024-10-24 | Leonardo DiCaprio Kate Winslet |
+-----+-----+
1 row in set (0,00 sec)
```

ii. **Listar título** de las **películas** en las que actúa **Leonardo DiCaprio**.

```
SELECT
ExtractValue(catalogo, '//Pelicula[Actores/Actor="Leonardo DiCaprio"]/Titulo') AS
titulo
FROM nuevos_catalogos;
```

```
ubuntu@mv-sgbd2: ~
mysql> SELECT
-> ExtractValue(catalogo, '//Pelicula[Actores/Actor="Leonardo DiCaprio"]/Titulo') AS titulo
-> FROM nuevos_catalogos;
+-----+
| titulo |
+-----+
| Titanic |
+-----+
1 row in set (0,00 sec)
```

iii. Obtener la duración de las películas de género Thriller.

```
SELECT
ExtractValue(catalogo, '//Pelicula[Genero="Thriller"]/Duracion') AS duracion
FROM nuevos_catalogos;
```

```
ubuntu@mv-sgbd2: ~
mysql> SELECT
-> ExtractValue(catalogo, '//Pelicula[Genero="Thriller"]/Duracion') AS duracion
-> FROM nuevos_catalogos;
+-----+
| duracion |
+-----+
| 106      |
+-----+
1 row in set (0,00 sec)
```

iv. Actualizar las películas en formato "VHS" a "Blue-Ray".

```
SELECT
    ExtractValue(catalogo, '//Pelicula[Formato="VHS"]/Titulo') AS Titulo,
    ExtractValue(catalogo, '//Pelicula[Formato="VHS"]/Formato') AS Formato
FROM nuevos_catalogos;
```

```
ubuntu@mv-sgbd2: ~
mysql> SELECT
-> ExtractValue(catalogo, '//Pelicula[Formato="VHS"]/Titulo') AS Titulo,
-> ExtractValue(catalogo, '//Pelicula[Formato="VHS"]/Formato') AS Formato
-> FROM nuevos_catalogos;
+-----+-----+
| Titulo          | Formato |
+-----+-----+
| The Sixth Sense | VHS     |
+-----+-----+
1 row in set (0,00 sec)
```

UpdateXML es una función busca en el **XML** el **nodo** que coincide con la ruta **XPath** proporcionada y reemplaza su contenido con el nuevo valor.

```
UPDATE nuevos_catalogos
SET
catalogo =
    UpdateXML(
        catalogo,
        '//Pelicula[Formato="VHS"]/Formato',
        '<Formato>Blue-Ray</Formato>')
WHERE catalogo LIKE '%<Formato>VHS</Formato>%';
```

```

ubuntu@mv-sgbd2: ~
mysql> UPDATE nuevos_catalogos
-> SET
-> catalogo =
-> UpdateXML(
-> catalogo,
->          '//Pelicula[Formato="VHS"]/Formato',
->          '<Formato>Blue-Ray</Formato>')
-> WHERE catalogo LIKE '%<Formato>VHS</Formato>%';
Query OK, 1 row affected (0,01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

```

```

SELECT
    ExtractValue(catalogo, '//Pelicula[Formato="Blue-Ray"]/Titulo') AS Titulo,
    ExtractValue(catalogo, '//Pelicula[Formato="Blue-Ray"]/Formato') AS Formato
FROM nuevos_catalogos;

```

```

ubuntu@mv-sgbd2: ~
mysql> SELECT
-> ExtractValue(catalogo, '//Pelicula[Formato="Blue-Ray"]/Titulo') AS Titulo,
-> ExtractValue(catalogo, '//Pelicula[Formato="Blue-Ray"]/Formato') AS Formato
-> FROM nuevos_catalogos;
+-----+-----+
| Titulo          | Formato |
+-----+-----+
| The Sixth Sense | Blue-Ray |
+-----+-----+
1 row in set (0,00 sec)

```

5) Instale la base de datos de ejemplo “world_x” y realice las siguientes acciones:

Una vez descargada la base de datos **world_x.sql**

Transferir **world_x.sql**:

```
multipass transfer C:\Users\criss\Documents\Ubuntu_Linux\world_x.sql  
mv-sgbd2:/home/ubuntu/world_x.sql
```

```
C:\ Símbolo del sistema  
C:\Users\criss>multipass transfer C:\Users\criss\Documents\Ubuntu_Linux\world_x.sql mv-sgbd2:/home/ubuntu/world_x.sql  
C:\Users\criss>
```

```
C:\ Seleccionar ubuntu@mv-sgbd2: ~  
ubuntu@mv-sgbd2:~$ ls -l  
total 33420  
-rw-rw-r-- 1 ubuntu ubuntu 20776686 oct  8 21:48 BinLog_1_sakila.sql  
-rw-rw-r-- 1 ubuntu ubuntu  7562117 oct  8 21:45 BinLog_2_sakila.sql  
-rw-rw-r-- 1 ubuntu ubuntu  1897106 oct  8 21:27 BinLog_sakila.sql  
-rw-rw-r-- 1 ubuntu ubuntu   18068 jul  3 11:28 mysql-apt-config_0.8.32-1_all.deb  
drwxrwxr-x 2 ubuntu ubuntu    4096 oct  8 18:59 sakila-db  
-rw-rw-r-- 1 ubuntu ubuntu  3388303 oct  8 19:42 sakila_FullBackup.sql  
-rwxrwxrwx 1 ubuntu ubuntu   558790 oct 22 21:44 world_x.sql  
ubuntu@mv-sgbd2:~$
```

Importar el esquema y crear la base de datos:

```
mysql -u root -p < /home/ubuntu/world_x.sql
```

```
ubuntu@mv-sgbd2: ~  
ubuntu@mv-sgbd2:~$ mysql -u root -p < /home/ubuntu/world_x.sql  
Enter password:  
ubuntu@mv-sgbd2:~$
```

```
USE world_x;
```

```
SHOW TABLES;
```

```
DESCRIBE countryinfo;
```

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sakila |
| sys |
| world_x |
+-----+
6 rows in set (0,00 sec)
```

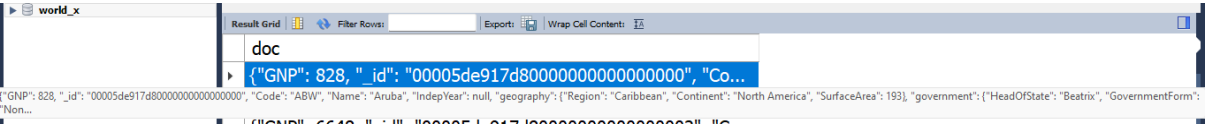
```
mysql> DESCRIBE countryinfo;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| doc | json | YES | | NULL | |
| _id | varbinary(32) | NO | PRI | NULL | STORED GENERATED |
| _json_schema | json | YES | | NULL | VIRTUAL GENERATED |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0,00 sec)
```

SELECT doc FROM countryinfo LIMIT 3;

SELECT JSON_PRETTY(doc) FROM countryinfo LIMIT 3;

```
mysql> SELECT doc FROM countryinfo LIMIT 3;
+-----+
| doc |
+-----+
| {"GNP": 828, "_id": "00005de917d80000000000000000", "Code": "ABW", "Name": "Aruba", "IndepYear": null, "geography": {"Region": "Caribbean", "Continent": "North America", "SurfaceArea": 193}, "government": {"HeadOfState": "Beatrix", "GovernmentForm": "Nonmetropolitan Territory of The Netherlands"}, "demographics": {"Population": 103000, "LifeExpectancy": 78.4000015258789}} |
| {"GNP": 5976, "_id": "00005de917d8000000000000000001", "Code": "AFG", "Name": "Afghanistan", "IndepYear": 1919, "geography": {"Region": "Southern and Central Asia", "Continent": "Asia", "SurfaceArea": 652090}, "government": {"HeadOfState": "Mohammad Omar", "GovernmentForm": "Islamic Emirate"}, "demographics": {"Population": 22720000, "LifeExpectancy": 45.90000152587896}} |
| {"GNP": 6648, "_id": "00005de917d8000000000000000002", "Code": "AGO", "Name": "Angola", "IndepYear": 1975, "geography": {"Region": "Central Africa", "Continent": "Africa", "SurfaceArea": 1246700}, "government": {"HeadOfState": "Jos\u00e9 Eduardo dos Santos", "GovernmentForm": "Republic"}, "demographics": {"Population": 12878000, "LifeExpectancy": 38.29999923706055}} |
+-----+
3 rows in set (0,00 sec)

mysql>
```




```
ubuntu@mv-sgbd2: ~
mysql> SELECT JSON_PRETTY(doc) FROM countryinfo LIMIT 3;
+-----+
| JSON_PRETTY(doc) |
+-----+
| {
  "GNP": 828,
  "_id": "00005de917d80000000000000000",
  "Code": "ABW",
  "Name": "Aruba",
  "IndepYear": null,
  "geography": {
    "Region": "Caribbean",
    "Continent": "North America",
    "SurfaceArea": 193
  },
  "government": {
    "HeadOfState": "Beatrix",
    "GovernmentForm": "Nonmetropolitan Territory of The Netherlands"
  },
  "demographics": {
    "Population": 103000,
    "LifeExpectancy": 78.4000015258789
  }
} |
| {
  "GNP": 5976,
  "_id": "00005de917d800000000000000001",
  "Code": "AFG",
  "Name": "Afghanistan",
  "IndepYear": 1919,
  "geography": {
    "Region": "Southern and Central Asia",
    "Continent": "Asia",
    "SurfaceArea": 652090
  },
  "government": {
    "HeadOfState": "Mohammad Omar",
    "GovernmentForm": "Islamic Emirate"
  },
  "demographics": {
    "Population": 22720000,
    "LifeExpectancy": 45.900001525878906
  }
} |
```

JSON_PRETTY(doc)

```
{ "GNP": 828, "_id": "00005de917d80000000000000000", "Code": "ABW", "Name": "Aruba", "Ind...
{ "GNP": 5976, "_id": "00005de917d800000000000000001", "Code": "AFG", "Name": "Afghanistan",...
{ "GNP": 6648, "_id": "00005de917d800000000000000002", "Code": "AGO", "Name": "Angola", "I...
```

JSON_PRETTY(doc)

```
{ "GNP": 828, "_id": { "GNP": 828,
  "Code": "ABW",
  "Name": "Aruba",
  "IndepYear": null,
  "geography": {
    "Region": "Caribbean",
    "Continent": "North America",
    "SurfaceArea": 193
  },
  "government": {
    "HeadOfState...
```

- a. Listar los **nombres** de las **ciudades** y su cantidad de habitantes.

```
select Name, json_extract(Info, "$.Population") as Population
from city
limit 10;
```

```
mysql> select Name, json_extract(Info, "$.Population") as Population from city limit 10;
+-----+-----+
| Name          | Population |
+-----+-----+
| Kabul         | 1780000    |
| Qandahar      | 237500     |
| Herat         | 186800     |
| Mazar-e-Sharif | 127800     |
| Amsterdam     | 731200     |
| Rotterdam     | 593321     |
| Haag          | 440900     |
| Utrecht       | 234323     |
| Eindhoven     | 201843     |
| Tilburg       | 193238     |
+-----+-----+
10 rows in set (0.00 sec)
```

- b. Encuentre las ciudades con una población superior a 1.000.000 de habitantes.

```
with citiesPopulation as (
  select Name, json_extract(Info, '$.Population') as Population
  from city
)

select *
from citiesPopulation c
where c.Population > 1000000;
```

```

ubuntu@mv-sgbd2: ~
mysql> with citiesPopulation as (
->   select Name, json_extract(Info, '$.Population') as Population
->   from city
-> )
->
-> select *
-> from citiesPopulation c
-> where c.Population > 1000000;
+-----+-----+
| Name                | Population |
+-----+-----+
| Kabul               | 1780000   |
| Alger               | 2168000   |
| Luanda              | 2022000   |
| Buenos Aires       | 2982146   |
| La Matanza          | 1266461   |
| Córdoba             | 1157507   |
| Yerevan             | 1248700   |
| Sydney              | 3276207   |
| Melbourne           | 2865329   |
| Brisbane            | 1291117   |
| Perth               | 1096829   |
| Baku                | 1787800   |
| Philadelphia        | 1517350   |
| Phoenix             | 1321045   |
| San Diego           | 1223400   |
| Dallas              | 1188580   |
| San Antonio         | 1144646   |
| Harare              | 1410000   |
+-----+-----+
237 rows in set (0,04 sec)

```

- c. **Mostrar** un listado “entendible” de la información registrada de los **10 últimos países ordenados** por código (campo **Code**).

```

select json_pretty(doc)
from countryinfo c
order by JSON_EXTRACT(c.doc, "$.Code") desc limit 10;

```

```

mysql> select json_pretty(doc) from countryinfo c order by JSON_EXTRACT(c.doc, "$.Code") desc limit 10;
+-----+
| json_pretty(doc) |
+-----+
| {                |
|   "GNP": 5951,    |
|   "_id": "00005de917d80000000000000000ee", |
|   "Code": "ZWE",  |
|   "Name": "Zimbabwe", |
|   "IndepYear": 1980, |
|   "geography": {  |
|     "Region": "Eastern Africa", |
|     "Continent": "Africa", |
|     "SurfaceArea": 390757 |
|   }, |
|   "government": { |
|     "HeadOfState": "Robert G. Mugabe", |
|     "GovernmentForm": "Republic" |
|   }, |
|   "demographics": { |
|     "Population": 11669000, |
|     "GNP": 5951, |
|     "GNP_per_capita": 511, |
|     "LifeExpectancy": 53, |
|     "Literacy": 75, |
|     "UrbanPop": 55, |
|     "FertilityRate": 4, |
|     "SexRatio": 100 |
|   } |
| } |
+-----+

```

Resultado:

```
+-----+
| json_pretty(doc) |
+-----+
| {
  "GNP": 5951,
  "_id": "00005de917d80000000000000000ee",
  "Code": "ZWE",
  "Name": "Zimbabwe",
  "IndepYear": 1980,
  "geography": {
    "Region": "Eastern Africa",
    "Continent": "Africa",
    "SurfaceArea": 390757
  },
  "government": {
    "HeadOfState": "Robert G. Mugabe",
    "GovernmentForm": "Republic"
  },
  "demographics": {
    "Population": 11669000,
    "LifeExpectancy": 37.79999923706055
  }
}
|
| {
  "GNP": 3377,
  "_id": "00005de917d80000000000000000ed",
  "Code": "ZMB",
  "Name": "Zambia",
  "IndepYear": 1964,
  "geography": {
    "Region": "Eastern Africa",
    "Continent": "Africa",
    "SurfaceArea": 752618
  },
  "government": {
    "HeadOfState": "Frederick Chiluba",
    "GovernmentForm": "Republic"
  },
  "demographics": {
    "Population": 9169000,
    "LifeExpectancy": 37.20000076293945
  }
}
|
| {
  "GNP": 116729,
  "_id": "00005de917d80000000000000000ec",
  "Code": "ZAF",
  "Name": "South Africa",
  "IndepYear": 1910,
  "geography": {
    "Region": "Southern Africa",
    "Continent": "Africa",
```

```

    "SurfaceArea": 1221037
  },
  "government": {
    "HeadOfState": "Thabo Mbeki",
    "GovernmentForm": "Republic"
  },
  "demographics": {
    "Population": 40377000,
    "LifeExpectancy": 51.099998474121094
  }
}
|
| {
  "GNP": 17000,
  "_id": "00005de917d800000000000000eb",
  "Code": "YUG",
  "Name": "Yugoslavia",
  "IndepYear": 1918,
  "geography": {
    "Region": "Southern Europe",
    "Continent": "Europe",
    "SurfaceArea": 102173
  },
  "government": {
    "HeadOfState": "Vojislav Kotunica",
    "GovernmentForm": "Federal Republic"
  },
  "demographics": {
    "Population": 10640000,
    "LifeExpectancy": 72.4000015258789
  }
}
|
| {
  "GNP": 6041,
  "_id": "00005de917d800000000000000ea",
  "Code": "YEM",
  "Name": "Yemen",
  "IndepYear": 1918,
  "geography": {
    "Region": "Middle East",
    "Continent": "Asia",
    "SurfaceArea": 527968
  },
  "government": {
    "HeadOfState": "Ali Abdallah Salih",
    "GovernmentForm": "Republic"
  },
  "demographics": {
    "Population": 18112000,
    "LifeExpectancy": 59.79999923706055
  }
}
|
| {
  "GNP": 141,

```

```

    "_id": "00005de917d800000000000000e9",
    "Code": "WSM",
    "Name": "Samoa",
    "IndepYear": 1962,
    "geography": {
      "Region": "Polynesia",
      "Continent": "Oceania",
      "SurfaceArea": 2831
    },
    "government": {
      "HeadOfState": "Malietoa Tanumafili II",
      "GovernmentForm": "Parlementary Monarchy"
    },
    "demographics": {
      "Population": 180000,
      "LifeExpectancy": 69.19999694824219
    }
  } |
  | {
    "GNP": 0,
    "_id": "00005de917d800000000000000e8",
    "Code": "WLF",
    "Name": "Wallis and Futuna",
    "IndepYear": null,
    "geography": {
      "Region": "Polynesia",
      "Continent": "Oceania",
      "SurfaceArea": 200
    },
    "government": {
      "HeadOfState": "Jacques Chirac",
      "GovernmentForm": "Nonmetropolitan Territory of France"
    },
    "demographics": {
      "Population": 15000,
      "LifeExpectancy": null
    }
  } |
  | {
    "GNP": 261,
    "_id": "00005de917d800000000000000e7",
    "Code": "VUT",
    "Name": "Vanuatu",
    "IndepYear": 1980,
    "geography": {
      "Region": "Melanesia",
      "Continent": "Oceania",
      "SurfaceArea": 12189
    },
    "government": {
      "HeadOfState": "John Bani",
      "GovernmentForm": "Republic"
    },
  },

```

```

    "demographics": {
      "Population": 190000,
      "LifeExpectancy": 60.599998474121094
    }
  }
  |
  | {
    "GNP": 21929,
    "_id": "00005de917d800000000000000e6",
    "Code": "VNM",
    "Name": "Vietnam",
    "IndepYear": 1945,
    "geography": {
      "Region": "Southeast Asia",
      "Continent": "Asia",
      "SurfaceArea": 331689
    },
    "government": {
      "HeadOfState": "Tr $\diamond$  Duc Luong",
      "GovernmentForm": "Socialistic Republic"
    },
    "demographics": {
      "Population": 79832000,
      "LifeExpectancy": 69.30000305175781
    }
  }
  |
  | {
    "GNP": 0,
    "_id": "00005de917d800000000000000e5",
    "Code": "VIR",
    "Name": "Virgin Islands, U.S.",
    "IndepYear": null,
    "geography": {
      "Region": "Caribbean",
      "Continent": "North America",
      "SurfaceArea": 347
    },
    "government": {
      "HeadOfState": "George W. Bush",
      "GovernmentForm": "US Territory"
    },
    "demographics": {
      "Population": 93000,
      "LifeExpectancy": 78.0999984741211
    }
  }
  |
+-----+

```

- d. Obtener las claves del documento **json** (campo **doc**) que se registran en **countryinfo**.

```
SELECT JSON_KEYS(doc) AS `keys`  
FROM countryinfo  
LIMIT 1;
```

```
ubuntu@mv-sgbd2: ~  
mysql> SELECT JSON_KEYS(doc) AS `keys`  
-> FROM countryinfo  
-> LIMIT 1;  
  
+-----+-----+  
| keys |  
+-----+-----+  
| ["GNP", "_id", "Code", "Name", "IndepYear", "geography", "government", "demographics"] |  
+-----+-----+  
1 row in set (0,00 sec)
```

- e. Analizar la estructura y contenidos del campo **doc** y agregar a la tabla **countryinfo** una **restricción CHECK** que permita validar los documentos de acuerdo a un esquema **json**. Comprobar el funcionamiento de la nueva restricción.

Análisis de la estructura del campo **doc:**

```
SELECT  
  json_type(doc->'$.GNP') AS GNP_type,  
  json_type(doc->'$_id') AS id_type,  
  json_type(doc->'$.Code') AS Code_type,  
  json_type(doc->'$.Name') AS Name_type,  
  json_type(doc->'$.IndepYear') AS IndepYear_type,  
  json_type(doc->'$.geography') AS geography_type,  
  json_type(doc->'$.government') AS government_type,  
  json_type(doc->'$.demographics') AS demographics_type  
FROM countryinfo LIMIT 1;
```

```
ubuntu@mv-sgbd2: ~  
mysql> SELECT  
->  json_type(doc->'$.GNP') AS GNP_type,  
->  json_type(doc->'$_id') AS id_type,  
->  json_type(doc->'$.Code') AS Code_type,  
->  json_type(doc->'$.Name') AS Name_type,  
->  json_type(doc->'$.IndepYear') AS IndepYear_type,  
->  json_type(doc->'$.geography') AS geography_type,  
->  json_type(doc->'$.government') AS government_type,  
->  json_type(doc->'$.demographics') AS demographics_type  
-> FROM countryinfo LIMIT 1;  
  
+-----+-----+-----+-----+-----+-----+-----+-----+  
| GNP_type | id_type | Code_type | Name_type | IndepYear_type | geography_type | government_type | demographics_type |  
+-----+-----+-----+-----+-----+-----+-----+-----+  
| INTEGER | STRING | STRING   | STRING   | NULL           | OBJECT        | OBJECT         | OBJECT            |  
+-----+-----+-----+-----+-----+-----+-----+-----+  
1 row in set (0,00 sec)
```


Definir el esquema JSON: El esquema especificado sigue las normas del borrador 07 de **JSON Schema** y establece los tipos de datos y campos requeridos.

- Campos obligatorios:
 - **GNP**
 - **_id**
 - **Code**
 - **Name**
 - **IndepYear**
 - **geography**
 - **government**
 - **demographics**
- Tipos de datos y restricciones adicionales, como el máximo de caracteres en **Code**.

```
SET @schema = '{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "GNP": {
      "type": "number"
    },
    "_id": {
      "type": "string"
    },
    "Code": {
      "type": "string",
      "maxLength": 3
    },
    "Name": {
      "type": "string"
    },
    "IndepYear": {
      "type": ["integer", "null"]
    },
    "geography": {
      "type": "object",
      "properties": {
        "Region": {
          "type": "string"
        },
        "Continent": {
          "type": "string"
        },
        "SurfaceArea": {
          "type": "number"
        }
      },
      "required": ["Region", "Continent", "SurfaceArea"]
    },
    "government": {
      "type": "object",
      "properties": {
```

```

        "HeadOfState": {
            "type": "string"
        },
        "GovernmentForm": {
            "type": "string"
        }
    },
    "required": ["HeadOfState", "GovernmentForm"]
},
"demographics": {
    "type": "object",
    "properties": {
        "Population": {
            "type": "integer"
        },
        "LifeExpectancy": {
            "type": ["number", "null"]
        }
    },
    "required": ["Population", "LifeExpectancy"]
}
},
"required": ["GNP", "_id", "Code", "Name", "IndepYear", "geography", "government",
"demographics"]
}';

```

```
SELECT doc INTO @document FROM countryinfo LIMIT 1;
```

```
SELECT JSON_SCHEMA_VALIDATION_REPORT(@schema, @document);
```

```

'>         "required": ["Population", "LifeExpectancy"]
'>     }
'> },
'>     "required": ["GNP", "_id", "Code", "Name", "IndepYear", "geogr
aphy", "government", "demographics"]
'> }';

```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql>
```

```
mysql>
```

```
mysql>
```

```
mysql> SELECT doc INTO @document FROM countryinfo LIMIT 1;
```

```
Query OK, 1 row affected (0.00 sec)
```

```
mysql>
```

```
mysql> SELECT JSON_SCHEMA_VALIDATION_REPORT(@schema, @document);
```

```
+-----+
```

```
| JSON_SCHEMA_VALIDATION_REPORT(@schema, @document) |
```

```
+-----+
```

```
| {"valid": true} |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

Para ver qué sucede si el **json** no es válido, modificamos el schema para agregar un patrón al id.

```
SET @schema = '{
  > "$schema": "http://json-schema.org/draft-07/schema#",
  > "type": "object",
  > "properties": {
  >   "GNP": {
  >     "type": "number"
  >   },
  >   "_id": {
  >     "type": "string",
  >     "pattern": "^[0-9a-f]{24}$"
  >   },
  ...

SELECT doc INTO @document FROM countryinfo LIMIT 1;

SELECT JSON_SCHEMA_VALIDATION_REPORT(@schema, @document);
```

```
mysql> SELECT doc INTO @document FROM countryinfo LIMIT 1;
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> SELECT JSON_SCHEMA_VALIDATION_REPORT(@schema, @document);
+-----+
| JSON_SCHEMA_VALIDATION_REPORT(@schema, @document) |
+-----+
| {"valid": false, "reason": "The JSON document location '#/_id' failed requirement 'pattern' at JSON Schema location '#/properties/_id'", "schema-location": "#/properties/_id", "document-location": "#/_id", "schema-failed-keyword": "pattern"} |
+-----+
1 row in set (0.00 sec)
```

f. **Listar** los **países** que no tienen registrado año de declaración de la independencia.

No podemos leer valores nulos con la función **json_extract**, pero sí podemos usar la función **json_value**:

```
SELECT
json_extract(doc, "$.IndepYear") AS indepYear, json_extract(doc, "$.Name") AS Name
FROM countryinfo
WHERE json_value(doc, "$.IndepYear") is null;
```

```
mysql> SELECT json_extract(doc, "$.IndepYear") AS indepYear,      json_extract(doc, "$.Name") AS Name
-> FROM countryinfo
-> WHERE json_value(doc, "$.IndepYear") is null;
+-----+-----+
| indepYear | Name                                     |
+-----+-----+
| null      | "Aruba"                                |
| null      | "Anguilla"                             |
| null      | "Netherlands Antilles"                 |
| null      | "American Samoa"                      |
| null      | "Antarctica"                           |
| null      | "French Southern territories"           |
| null      | "Bermuda"                              |
| null      | "Bouvet Island"                        |
| null      | "Cocos (Keeling) Islands"              |
| null      | "Cook Islands"                         |
| null      | "Christmas Island"                     |
| null      | "Cayman Islands"                       |
| null      | "Western Sahara"                       |
| null      | "Falkland Islands"                     |
| null      | "Faroe Islands"                        |
| null      | "Gibraltar"                            |
| null      | "Guadeloupe"                           |
| null      | "Greenland"                            |
| null      | "French Guiana"                         |
| null      | "Guam"                                  |
| null      | "Hong Kong"                            |
| null      | "Heard Island and McDonald Islands"    |
| null      | "British Indian Ocean Territory"       |
| null      | "Macao"                                 |
| null      | "Northern Mariana Islands"             |
| null      | "Montserrat"                           |
| null      | "Martinique"                           |
| null      | "Mayotte"                               |
| null      | "New Caledonia"                        |
| null      | "Norfolk Island"                       |
| null      | "Niue"                                  |
| null      | "Pitcairn"                             |
| null      | "Puerto Rico"                         |
| null      | "Palestine"                            |
| null      | "French Polynesia"                     |
| null      | "Rönion"                                |
| null      | "South Georgia and the South Sandwich Islands" |
| null      | "Saint Helena"                         |
| null      | "Svalbard and Jan Mayen"               |
| null      | "Saint Pierre and Miquelon"            |
| null      | "Turks and Caicos Islands"             |
| null      | "Tokelau"                              |
| null      | "East Timor"                           |
| null      | "United States Minor Outlying Islands" |
| null      | "Virgin Islands, British"              |
| null      | "Virgin Islands, U.S."                 |
| null      | "Wallis and Futuna"                    |
+-----+-----+
47 rows in set (0.00 sec)
```

- g. Listar los datos demográficos y la población de su capital para los 10 países con menor superficie.

```
SELECT
  CountryName,
  Population AS CountryPopulation,
  SurfaceArea AS "SurfaceArea (km^2)",
  ci.Name AS CapitalName,
  JSON_EXTRACT(Info, "$.Population") AS CapitalPopulation,
  LifeExpectancy
FROM (
  SELECT
    JSON_UNQUOTE(JSON_EXTRACT(doc, "$.Name")) AS CountryName,
    JSON_EXTRACT(doc, "$.demographics.Population") AS Population,
    JSON_EXTRACT(doc, "$.geography.SurfaceArea") AS SurfaceArea,
    JSON_EXTRACT(doc, "$.demographics.LifeExpectancy") AS LifeExpectancy
  FROM countryinfo c
) AS table1
JOIN country c ON table1.CountryName = c.Name
JOIN city ci ON ci.ID = c.Capital
ORDER BY SurfaceArea ASC
LIMIT 10;
```

```
mysql> SELECT
->   CountryName,
->   Population AS CountryPopulation,
->   SurfaceArea AS "SurfaceArea (km^2)",
->   ci.Name AS CapitalName,
->   JSON_EXTRACT(Info, "$.Population") AS CapitalPopulation,
->   LifeExpectancy
-> FROM (
->   SELECT
->     JSON_UNQUOTE(JSON_EXTRACT(doc, "$.Name")) AS CountryName,
->     JSON_EXTRACT(doc, "$.demographics.Population") AS Population,
->     JSON_EXTRACT(doc, "$.geography.SurfaceArea") AS SurfaceArea,
->     JSON_EXTRACT(doc, "$.demographics.LifeExpectancy") AS LifeExpectancy
->   FROM countryinfo c
-> ) AS table1
-> JOIN country c ON table1.CountryName = c.Name
-> JOIN city ci ON ci.ID = c.Capital
-> ORDER BY SurfaceArea ASC
-> LIMIT 10;
```

CountryName	CountryPopulation	SurfaceArea (km^2)	CapitalName	CapitalPopulation	LifeExpectancy
Holy See (Vatican City State)	1000	0.4000000059604645	Città del Vaticano	455	null
Monaco	34000	1.5	Monaco-Ville	1234	78.80000305175781
Gibraltar	25000	6	Gibraltar	27025	79
Tokelau	2000	12	Fakaofu	300	null
Cocos (Keeling) Islands	600	14	West Island	167	null
Macao	473000	18	Macao	437500	81.5999984741211
Nauru	12000	21	Yaren	559	60.79999923706055
Tuvalu	12000	26	Funafuti	4600	66.30000305175781
Norfolk Island	2000	36	Kingston	800	null
Pitcairn	50	49	Adamstown	42	null

10 rows in set (0.05 sec)

- h. **Agregar una clave** al campo **Info** de la tabla **city** para representar el **código de área telefónico** 362 de la ciudad de Resistencia.

```
UPDATE city
SET Info = JSON_SET(Info, "$.codArea", null);
```

```
mysql> UPDATE city SET Info = JSON_SET(Info, "$.codArea", null);
Query OK, 4079 rows affected (0.22 sec)
Rows matched: 4079  Changed: 4079  Warnings: 0

mysql> select * from city limit 10;
+----+-----+-----+-----+-----+
| ID | Name       | CountryCode | District | Info                                     |
+----+-----+-----+-----+-----+
| 1  | Kabul      | AFG         | Kabul    | {"codArea": null, "Population": 1780000} |
| 2  | Qandahar   | AFG         | Qandahar | {"codArea": null, "Population": 237500}  |
| 3  | Herat      | AFG         | Herat    | {"codArea": null, "Population": 186800}  |
| 4  | Mazar-e-Sharif | AFG        | Balkh    | {"codArea": null, "Population": 127800}  |
| 5  | Amsterdam  | NLD         | Noord-Holland | {"codArea": null, "Population": 731200} |
| 6  | Rotterdam  | NLD         | Zuid-Holland | {"codArea": null, "Population": 593321}  |
| 7  | Haag       | NLD         | Zuid-Holland | {"codArea": null, "Population": 440900}  |
| 8  | Utrecht    | NLD         | Utrecht  | {"codArea": null, "Population": 234323}  |
| 9  | Eindhoven  | NLD         | Noord-Brabant | {"codArea": null, "Population": 201843} |
| 10 | Tilburg    | NLD         | Noord-Brabant | {"codArea": null, "Population": 193238} |
+----+-----+-----+-----+-----+
10 rows in set (0.00 sec)
```

En **MySQL**, las comillas dobles (" ") pueden ser usadas para delimitar cadenas de texto, pero es más común y recomendable usar comillas simples (' ') para valores de texto en las consultas **SQL**.

```
UPDATE city
SET Info = JSON_SET(Info, "$.codArea", 362)
where Name = "Resistencia";
```

```
mysql> UPDATE city SET Info = JSON_SET(Info, "$.codArea", 362) where Name = "Resistencia";
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * from city where Name = "Resistencia";
+----+-----+-----+-----+-----+
| ID | Name       | CountryCode | District | Info                                     |
+----+-----+-----+-----+-----+
| 98 | Resistencia | ARG         | Chaco    | {"codArea": 362, "Population": 229212} |
+----+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

- i. **Listar** la información de los **países** cuya **forma de gobierno** es **república federal** ("GovernmentForm": "Federal Republic").

```
select json_pretty(doc)
from countryinfo c
where json_value(c.doc, "$.government.GovernmentForm") = "Federal Republic";
```

```
mysql> select json_pretty(doc) from countryinfo c where json_value(c.doc, "$.government.GovernmentForm") = "Federal Republic";
+-----+
| json_pretty(doc) |
+-----+
| {
  "GNP": 340238,
  "_id": "00005de917d8000000000000000008",
  "Code": "ARG",
  "Name": "Argentina",
  "IndepYear": 1816,
  "geography": {
    "Region": "South America",
    "Continent": "South America",
    "SurfaceArea": 2780400
  },
  "government": {
    "HeadOfState": "Carlos Menem",
    "GovernmentForm": "Federal Republic"
  },
  "demographics": {
    "Population": 3400000,
    "LifeExpectancy": 75.4
  }
}
```

```
| {
  "GNP": 17000,
  "_id": "00005de917d80000000000000000eb",
  "Code": "YUG",
  "Name": "Yugoslavia",
  "IndepYear": 1918,
  "geography": {
    "Region": "Southern Europe",
    "Continent": "Europe",
    "SurfaceArea": 102173
  },
  "government": {
    "HeadOfState": "Vojislav Kotunica",
    "GovernmentForm": "Federal Republic"
  },
  "demographics": {
    "Population": 10640000,
    "LifeExpectancy": 72.4
  }
}
+-----+
15 rows in set (0.01 sec)
```


- j. **Actualizar la población de Argentina** a 46.044.703 habitantes.

```
SELECT JSON_PRETTY(doc)
FROM countryinfo
WHERE JSON_VALUE(doc, "$.Name") = 'Argentina';
```

```
C:\> Seleccionar ubuntu@mv-sgbd2: ~
mysql> SELECT JSON_PRETTY(doc)
      -> FROM countryinfo
      -> WHERE JSON_VALUE(doc, "$.Name") = 'Argentina';
+-----+
| JSON_PRETTY(doc) |
+-----+
| {
  "GNP": 340238,
  "_id": "00005de917d800000000000000000008",
  "Code": "ARG",
  "Name": "Argentina",
  "IndepYear": 1816,
  "geography": {
    "Region": "South America",
    "Continent": "South America",
    "SurfaceArea": 2780400
  },
  "government": {
    "HeadOfState": "Fernando de la Rúa",
    "GovernmentForm": "Federal Republic"
  },
  "demographics": {
    "Population": 37032000,
    "LifeExpectancy": 75.0999984741211
  }
} |
+-----+
1 row in set (0,00 sec)
```

```
UPDATE countryinfo
SET doc = JSON_SET(doc, "$.demographics.Population", 46044703)
WHERE JSON_VALUE(doc, "$.Name") = 'Argentina';
```

```
C:\> ubuntu@mv-sgbd2: ~
mysql> UPDATE countryinfo
      -> SET doc = JSON_SET(doc, "$.demographics.Population", 46044703)
      -> WHERE JSON_VALUE(doc, "$.Name") = 'Argentina';
Query OK, 1 row affected (0,05 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```



```
select json_pretty(doc) from countryinfo c where json_extract(doc, "$.Name") = 'Argentina';
```

```
mysql> select json_pretty(doc) from countryinfo c where json_extract(c.doc, "$.Name") = 'Argentina';
+-----+
| json_pretty(doc) |
+-----+
| {
  "GNP": 340238,
  "_id": "00005de917d8000000000000000008",
  "Code": "ARG",
  "Name": "Argentina",
  "IndepYear": 1816,
  "geography": {
    "Region": "South America",
    "Continent": "South America",
    "SurfaceArea": 2780400
  },
  "government": {
    "HeadOfState": "Fernando de la Ra",
    "GovernmentForm": "Federal Republic"
  },
  "demographics": {
    "Population": 46044703,
    "LifeExpectancy": 75.0999984741211
  }
} |
+-----+
1 row in set (0.00 sec)
```